ARCOS – Orchestrator (Maestro) Guide

1. Purpose of this Guide

This guide explains the role of the ARCOS Orchestrator (Maestro). It is written for developers and integrators, clarifying that the Orchestrator is a purely schema-driven coordinator. It is completely agnostic of domain specifics and focuses only on sequencing, routing, and validating XML messages between agents using schemas.

2. Role in ARCOS

The Orchestrator is the central traffic controller. It:
- Initializes projects and manages their lifecycle.
- Sends and receives XML messages according to schemas.
- Ensures every exchange between agents (Speculus, Producer, Validator, Post-Processor, Filter) is valid against its XSD.
- Handles retries, error recovery, and loop control.

The Orchestrator itself has no business logic — it only manages communication and enforces contracts. It loops until success or max trials.

3. Responsibilities

- Manage project state (IDs, lifecycles).
- Dispatch requests to appropriate agents (ARCOS Speculus, Domain Speculus, Producer, Validator, Post-Processor, Filter).
- Validate all XML messages against their schemas before forwarding.
- Handle retry cycles with Filter involvement when errors occur.
- Collect reports and deliver final output.zip to the user.
- Ensure full traceability: link every report and artifact to a ProjectID.

4. Workflow

1. Project Init: Receives a new project or edit request, launches ARCOS Speculus and produce ARCOS_Project.xml.
2. Rule Extraction: Sends to Domain Speculus, receives Domain_Rules.xml.
3. Production: Calls Producer with ProjectSpec, DomainSchema, Rules, and on retry, filtered output.zip and report.
4. Validation: Sends Producer output to Validator → gets ValidatorReport.xml.
5. Post-Processing: Sends validated outputs to Post-Processor → gets PostProcessorReport.xml.
6. Retry (if needed): If errors occur, Orchestrator calls Filter → trimmed output.zip + reports → Producer retry.
7. Final Verification: Once success, Orchestrator rebuilds full output.zip, validates, post-processes, then delivers to the user.

5. Inputs and Outputs

Inputs:
- ARCOS_Project.xml as the ARCOS-Speculus response
- Domain Speculus response as Domain_Rules.xml
- Producer responses with output.zip
- Validator reports & responses
- Post-Processor reports & responses
- Filter responses

Outputs:

- Structured logs of all schema-validated interactions.
- Final validated output.zip delivered to the user.
- Audit trail (reports + clarifications + safe store references).

6. Example (BLEU Inventory Domain)

In BLEU, the Orchestrator:

1. Starts a project referencing bleu_parts_v5.xsd.
2. Routes rules from Domain Speculus.
3. Sends them with project spec to Producer → Rust CRUD code.
4. Validates code via Validator → compliance checks.
5. If Validator fails, filters the codebase → retry Producer.
6. On success, Post-Processor compiles, report is collected.
7. Orchestrator returns a clean, validated output.zip with reports.

7. Benefits

- Domain-agnostic: never interprets rules itself.
- Reliable: ensures that every step passes schema validation.
- Traceable: audit logs connect inputs, outputs, and rules.
- Extensible: adding new agents or domains requires no changes to the Orchestrator itself.
- Error-resilient: built-in retry and filtering loop.

8. Conclusion

The Orchestrator is the backbone of ARCOS. By remaining neutral and enforcing only schema contracts, it enables complex multi-agent workflows to execute consistently. Its strength lies in coordination, validation, and traceability — never in domain logic.